UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/765,827	01/26/2004	Yolanta Beresnevichiene	20020754 2-2	9077

22879 7590 12/08/2010

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
3404 E. Harmony Road
Mail Stop 35
FORT COLLINS, CO 80528

EXAMINER	
CAO, DIEM K	

ART UNIT	PAPER NUMBER
2196	

NOTIFICATION DATE	DELIVERY MODE
12/08/2010	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

JERRY.SHORMA@HP.COM
ipa.mail@hp.com
laura.m.clark@hp.com

Office Action Summary	Application No.	Applicant(s)	
	10/765,827	BERESNEVICHIENE ET AL.	
	Examiner	Art Unit	
	DIEM K. CAO	2196	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *19 November 2010*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-3,5-19,21-37 and 39-59* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-3,5-19,21-37 and 39-59* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-3, 5-19, 21-37 and 39-59 are pending. Applicant has amended claims 1, 11,-18, 21, 23, 25, 29-35, 43-50, 54-56, and 58-59 and canceled claims 4 and 20.

Continued Examination Under 37 CFR 1.114

2. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 11/19/2010 has been entered.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1-3, 5-14, 18-19, 21-32, 34-37, 39-53, 58 and 59 are rejected under 35 U.S.C. 103(a) as being unpatentable over Van Dyke et al. (US 6,412,070 B1) in view of Larus et al. (EEL: Machine-Independent Executable Editing).

As to claim 1, Van Dyke teaches a method of computer operating system data management (abstract) comprising the steps of:

associating data management information with data input to a process (Each application 140 issues an access request to operating system 120 when desiring to operate on one of the computing objects 125; col. 5, lines 18-20 and Upon receiving access request, operating system examines the security information for each object; col. 5, lines 30-31 and col. 7, lines 28-36); and

regulating operating system operations involving the data according to the data management information (determines whether application … operating system 120 enforces access request 150; col. 5, lines 32-35 and col. 7, lines 38-44); and

associate first data management information with a first subset of the data, to associate second data management information with a second subset of the data (col. 12, lines 1-37), and to verify that the data management information indicates that the data is authorized to be written by an instruction to write the data before the data is written (col. 8, line 55 – col. 9, line 20).

Van Dyke does not teach disassembling an application to be executed to obtain machine code; and modifying the obtained machine code of the application to include instructions to associate first data management information with a first subset of the data, to associate second data management information with a second subset of the data, and to verify that the data management information indicates that the data is authorized to be written by an instruction to write the data before the data is written.

However, Larus teaches disassembling an application to be executed to obtain machine code; and modifying the obtained machine code of the application to include instructions to observe, measure or modify a program behavior (EEL, by contract, directly analyzes and modifies a program's instructions, and consequently can operate on program without relocation

information, such as fully compiled and linked programs; pages 292, right column, fourth paragraph and section 3.3 Control-Flow Graph; pages 294-295).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to apply the teaching of Larus to the system of Van Dyke because by adding the data management information to the program itself, it would improve the performance of the system/application at runtime by without checking the data management information when encounting the data.

As to claim 2, Van Dyke teaches wherein supervisor code administers the method by controlling the process at runtime (Upon receiving access request, operating system examines the security information for each object; col. 5, lines 30-31 and col. 7, lines 28-36).

As to claim 3, Van Dyke teaches wherein associating the data management information with the data input to the process comprises associating the data management information with the data as the data is read into a memory space (Each application 140 issues an access request to operating system 120 when desiring to operate on one of the computing objects 125; col. 5, lines 18-20).

As to claim 5, Van Dyke teaches wherein associating the data management information with the data input to the process comprises associating the data management information with each independently addressable data unit that is read into the memory space (each object is associated with a security descriptor … does not allow any access rights; col. 6, lines 25-44).

As to claim 6, Van Dyke teaches wherein the data management information is written to a data management memory space under control of the supervisor code (operating system 120 retrieves the security descriptor for the requested object 125. Next, operating system 120 examines the access control list; col. 8, lines 59-61).

As to claim 7, Van Dyke teaches wherein the supervisor code comprises state machine automatons arranged to control the writing of the data management info to the data management memory space (inherent from the operating system retrieves the security descriptor for the requested object upon receiving the access request from application; col. 8, lines 55-61).

As to claim 8, Van Dyke teaches wherein regulating the operating system operation comprises:

identifying an operation involving the data (inherent from "When application issues access request … desired operation on object 125"; col. 6,lines 18-24, thus, which operation must be identified);

if the operation involves the data and is carried out within the process, maintaining an association between an output of the operation and the data management information (computing objects represent objects defined by Applications 140 … address space; col. 5, lines 8-16 and col. 6, lines 25-28); and

if the operation involving the data includes a write operation to a location external to the process (a command to read or write a file; col. 5, line 27), selectively performing the operation

dependent on the data management information (operating system … enforces access request 150; col. 5, lines 30-35 and col. 6, lines 36-44).

As to claim 9, Van Dyke teaches wherein identifying the operation comprises: analyzing process instructions to identify the operation involving the data (inherent from "When application issues access request … desired operation on object 125"; col. 6, lines 18-24, thus, which operation must be identified), and providing instructions relating to the data management information with the operation involving the data (col. 6, lines 36-44).

As to claim 10, Van Dyke as modified by Larus teaches wherein the process instructions are analysed as blocks, each block defined by operations up to a terminating condition (see Larus: page 293, right paragraph).

As to claim 11, Van Dyke as modified by Larus teaches wherein code of an application is analyzed statically in order to create a control flow graph (see Larus: page 294, section 3.3).

As to claim 12, Van Dyke as modified by Larus teaches wherein the code is analysed before load time (see Larus: inherent since the system verifies the program before runtime.).

As to claim 13, Van Dyke does not teach wherein the code is analysed at load time. However, it would have been obvious to one of ordinary skill in the art that the teaching of Larus can be applied to verify the program before load time and at load time.

As to claim 14, Van Dyke as modified by Larus teaches wherein code of an application is instrumented to identify an entry point of a conditional structure in the code and an exit point of the conditional structure, and in which the entry points and exits points are identified from the control flow graph (see Larus: pages 294-295).

As to claim 18, it is the same as the method claim of claim 1 except this is a computing platform claim, and is rejected under the same ground of rejection.

As to claim 19, Van Dyke teaches a memory space, the computing platform arranged to load the process into the memory space and run the process under control of the data management unit (col. 4, lines 7-11).

As to claim 21, see rejection of claim 5 above.

As to claim 22, Van Dyke teaches wherein the data management unit comprises part of an operating system kernel space (operating system 120; col. 4, lines 7-11).

As to claim 23, Van Dyke teaches wherein the operating system kernel space comprises a tagging driver arranged to control loading of a supervisor code into the memory space with the process (col. 3, lines 50-59).

As to claim 24, see rejection of claim 2 above.

As to claim 25, see rejection of claim 9 above.

As to claim 26, see rejection of claim 6 above.

As to claim 27, Van Dyke teaches wherein the data management unit comprises a data filter arranged to identify data management information associated with data that is to be read into the memory space (operating system grants or denies access requests 150 based on access mask 210 … corresponding object 125; col. 6, lines 59-64).

As to claim 28, Van Dyke teaches wherein the data filter is arranged to associate data management information with data read into the memory space from predetermined sources, or alternatively is arranged to associated default data management information with data read into the memory space (The definition … directory service; col. 6, lines 4-10 and col. 8, lines 55-65).

As to claim 29, Van Dyke teaches wherein the data management unit further comprises a tag management module arranged to allow a user to specify data management information to be associated with data (col. 9, lines 36-59).

As to claim 30, Van Dyke teaches wherein the data management unit comprises a tag propagation module arranged to maintain an association with the data that has been read into the

process and the data management information associated therewith (col. 8, lines 55-65. Although Van Dyke does not explicitly use the term "a tag propagation module", the operating system performs the same function as a tag propagation module.).

As to claim 31, Van Dyke teaches wherein the tag propagation module is arranged to maintain an association between an output of operations carried out within the process and the data management information associated with the data involved in the operations (col. 10, lines 6-19).

As to claim 32, Van Dyke teaches wherein the tag propagation module comprises state machine automatons arranged to maintain an association between an output of operations carried out within the process and the data management information associated with the data involved in the operations (col. 9, lines 41-59).

As to claim 34, Van Dyke teaches an operating system data management method (abstract) comprising the step of: identifying data having data management information associated therewith when the data is to be read into a memory space (Each application 140 issues an access request to operating system 120 when desiring to operate on one of the computing objects 125; col. 5, lines 18-20 and Upon receiving access request, operating system examines the security information for each object; col. 5, lines 30-31 and col. 7, lines 28-36), associate first data management information with a first subset of the data, to associate second data management information with a second subset of the data (col. 12, lines 1-37), and to verify

that the data management information indicates that the data is authorized to be written by an instruction to write the data before the data is written (col. 8, line 55 – col. 9, line 20).

Van Dyke does not teach disassembling an application to be executed to obtain machine code; and modifying the obtained machine code of the application to include instructions to identify the data.

However, Larus teaches disassembling an application to be executed to obtain machine code; and modifying the obtained machine code of the application to include instructions to observe, measure or modify a program behavior (EEL, by contract, directly analyzes and modifies a program's instructions, and consequently can operate on program without relocation information, such as fully compiled and linked programs; pages 292, right column, fourth paragraph and section 3.3 Control-Flow Graph; pages 294-295).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to apply the teaching of Larus to the system of Van Dyke because by adding the data management information to the program itself, it would improve the performance of the system/application at runtime by without checking the data management information when encounting the data.

As to claim 35, Van Dyke teaches associating data management information with the data in response to determining that no data management information associated with the data (col. 6, lines 18-24).

As to claim 36, Van Dyke teaches wherein the data management information associated with data is read into the memory space with the data (operating system 120 retrieves the security descriptor for the requested object 125. Next, operating system 120 examines the access control list; col. 8, lines 59-61).

As to claim 37, Van Dyke teaches maintaining an association between the data and the data management information when the data is involved in operations within the process (col. 6, lines 18-24), and associating data management information with other data resulting from operations involving the data (col. 5, lines 42-44 and col. 6, lines 25-27).

As to claim 39, Van Dyke teaches examining the data management information when the data is to be involved in a operation external to the process, and allowing the operation if it is compatible with the data management information (col. 8, line 55 – col. 9, line 2).

As to claim 40, Van Dyke teaches wherein the operation is blocked if it is not compatible with the data management information (col. 6, lines 41-44).

As to claim 41, Van Dyke teaches wherein the operation external to the process is compatible with the data management information subject to including the associated data management information with an output of the operation (col. 5, lines 2-7, lines 30-35).

As to claim 42, Van Dyke teaches wherein the data management information identifies a set of permitted operations (col. 5, lines 39-44).

As to claim 43, see rejection of claim 1 above. Van Dyke further teaches a processor to identify data having data management information associated therewith when that data is read into a memory space (Each application 140 issues an access request to operating system 120 when desiring to operate on one of the computing objects 125; col. 5, lines 18-20 and Upon receiving access request, operating system examines the security information for each object; col. 5, lines 30-31 and col. 7, lines 28-36. It is noted that although Van Dyke does not use the term "a data filter", the operating system of Van Dyke performs the same functionality of the data filter.).

As to claim 44, Van Dyke teaches wherein the processor is to associate data management information with the data if the data is identified as having no data management information associated therewith (The definition … directory service; col. 6, lines 4-10 and col. 8, lines 55-65).

As to claim 45, Van Dyke teaches wherein the processor is to read the data management information associated with the data into the memory space with the data (col. 8, lines 59-60).

As to claim 46, see rejection of claim 37 above.

As to claim 47, see rejection of claim 38 above. It is noted that although Van Dyke does not use the term "tag propagation module", the operating system of Van Dyke performs the same functionality of the "tag propagation module".

As to claim 48, Van Dyke teaches wherein the tag propagation module is to examine the data management information when the data is to be involved in an operation external to the process, and to cause the operation to be allowed if it is compatible with the data management information (col. 8, line 55 – col. 9, line 2).

As to claim 49, Van Dyke teaches wherein the tag propagation module is to cause the operation to be blocked if the operation is not compatible with the data management information (col. 6, lines 41-44).

As to claim 50, Van Dyke teaches wherein the tag propagation module is arranged to perform the operation external to the process subject to including the associated data management information with an output of the operation (col. 5, lines 2-7, lines 30-35).

As to claim 51, Van Dyke teaches wherein the data management information identifies a set of permitted operations (col. 5, lines 39-44).

As to claim 52, it is the same as the method claim of claim 1 except this a computer program, and is rejected under the same ground of rejection.

As to claim 53, this is the same as the method claim of claim 31 except this is a computer program claim, and is rejected under the same ground of rejection.

As to claim 58, it is the same as the method of claim 1 except this is an operating system claim, and is rejected under the same ground of rejection.

As to claim 59, it is the same as the method of claim 34 except this is an operating system claim, and is rejected under the same ground of rejection.

5. Claims 15-17 and 33 are rejected under 35 U.S.C. 103(a) as being unpatentable over Van Dyke et al. (US 6,412,070 B1) in view of Besson et al. (Model checking security properties of control flow graphs) further in view of Larus et al. (EEL: Machine-Independent Executable Editing).

As to claim 15, Van Dyke does not teach in which the conditional structure includes a conditional expression, a process has a tag associated with a program counter stack and when the entry point of a conditional structure is identified at run-time, a current tag is pushed further on the program counter stack, and a new tag associated with the conditional expression is added to the front of the counter stack. However, Besson teaches in which the conditional structure includes a conditional expression, a process has a tag associated with a program counter stack and when the entry point of a conditional structure is identified at run-time, a current tag is

pushed further on the program counter stack, and a new tag associated with the conditional expression is added to the front of the counter stack (pages 4-5, section 2.1 and page 9, section "Stack inspection"). It would have been obvious to one of ordinary skill in the art at the time the invention was made to apply the teaching of Besson to the system of Van Dyke because Besson teaches a minimalistic, security-dedicated program model that only contains procedure and run-time security checks and propose an automatic method for verifying that an implementation using local security checks satisfies a global security property (abstract).

As to claim 16, Van Dyke as modified by Besson teaches in which when the exit point of a conditional structure is identified at run time, the tag from the entry point of the conditional structure is returned to the front of the counter stack (page 4, last paragraph).

As to claim 17, Van Dyke as modified by Besson teaches in which during all operations from an entry of the conditional structure, the tags of the locations in branching expressions are updated according to the tag of the program counter stack (pages 9-10, section "stack inspection").

As to claim 33, Van Dyke as modified by Besson teaches in which code of an application is instrumented to identify an entry point of a conditional structure in the code and an exit point of the conditional structure, the computer platform further comprises a static code analyzer to identify conditional branch entry and exit points and a conditional tag propagator for run-time

propagation of tags associated with data storage locations included in the conditional structure (see Besson: pages 3-4, section "Program Model").

6. Claims 54- 57 are rejected under 35 U.S.C. 103(a) as being unpatentable over Besson et al. (Model checking security properties of control flow graphs) in view of Larus et al. (EEL: Machine-Independent Executable Editing) further in view of Van Dyke et al. (US 6,412,070 B1).

As to claim 54, Besson teaches a method of modifying computer code of an application, the method comprising the steps of identifying conditional branches in code and instrumenting the code to provide information regarding the entry and exit points of the conditional structures (pages 2-3, section 2 "Program model" and each piece of code is granted a set of permissions to execute certain operations; page 9, section Stack inspection).

Besson does not teach modifying the machine code to include instructions that, when executed, cause a computer to regulate data according to data management information, wherein the instructions to regulate the data according to the data management information include instructions to associate first data management information with a first subset of the data and second data management information with a second subset of the data and to verify that the data management information indicates that the data is authorized to be written by an instruction to write the data before the data is written.

However, Larus teaches modifying the machine code to include instructions to observe, measure or modify a program behavior (EEL, by contract, directly analyzes and modifies a

program's instructions, and consequently can operate on program without relocation information, such as fully compiled and linked programs; pages 292, right column, fourth paragraph and section 3.3 Control-Flow Graph; pages 294-295). It would have been obvious to one of ordinary skill in the art at the time the invention was made to apply the teaching of Larus to the system of Besson because by adding the data management information to the program itself, it would improve the performance of the system/application at runtime by without checking the data management information when encounting the data.

Van Dyke teaches associate first data management information with a first subset of the data, to associate second data management information with a second subset of the data (col. 12, lines 1-37), and to verify that the data management information indicates that the data is authorized to be written by an instruction to write the data before the data is written (col. 8, line 55 – col. 9, line 20). It would have been obvious to one of ordinary skill in the art at the time the invention was made to apply the teaching the Van Dyke to the system of Besson because Van Dyke teaches a method to extending access control to resource in the system in different level.

As to claim 55, Besson teaches in which the modification is carried out before the load time (inherent since the system verify the program before runtime).

As to claim 56, Besson does not teach in which the modification is carried out at load time. However, it would have been obvious to one of ordinary skill in the art that the teaching of Besson can be applied to verify the program before load time and at load time.

As to claim 57, Besson teaches creating a control flow graph representing of the code and analyzing the conditional flow graph to identify conditional branches in the code (page 20, section Constructing the control flow graph).

Response to Arguments

7. Applicant's arguments filed 11/19/2010 have been fully considered but they are not persuasive.

In response to Applicant's argument that Van Dyke and Larus and Besson do not teach modifying the application to include instruction to associate first data management information with a first subset of the data and second data management information with a second subset of the data and to verify that the data management information indicates that the data is authorized to be written by an instruction to write the data before the data is written, the rejection has been clarified to show that Van Dyke as modified teaches the above limitation.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to DIEM K. CAO whose telephone number is (571)272-3760. The examiner can normally be reached on Monday - Thursday, 8:00AM - 5:00PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Emerson Puente can be reached on (571) 272-3652. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/DIEM K CAO/
Primary Examiner
Art Unit 2196

DC
December 3, 2010